课后作业一：多项式回归模型

任务：多项式回归是一种回归分析形式，在这种形式中，自变量（x）和因变量（y）之间的关系被建模为（n）阶多项式。使用机器学习的方法来创建一个多项式回归模型，该模型可以根据给定的数据集预测结果。数据集由自变量（x）和因变量（y）组成，你的任务是找到一个多项式，能最好地描述（x）和（y）之间的关系。

数据集：本实验选取数据集包含 125 个样本点，每个样本具有一个自变量(x)和一个因变量(y)。数据集根据 4:1 的比例划分为训练集和测试集。

实现：

1.导入必要的库

```
import pandas as pd    #用来分析结构化数据
import numpy as np     #提供高性能的矩阵运算
import matplotlib.pyplot as plt  #可视化工具
import csv  #读写文件的库
```

2.函数定义

（1）函数 f(x, theta)：用于计算输入 x 和参数 theta 的多项式回归函数值。其中 x 是一个矩阵，每一行对应一个输入样本；theta 是一个参数向量。

```
def f(x,theta):
    return np.dot(x,theta)
```

（2）函数 standardize(x)：用于对输入数据 x 进行标准化，通过减去均值并除以标准差。

```
def standardize(x):
    return (x - np.mean(x)) / np.std(x)
```

（3）函数 to_matrix(x)：用于将输入向量 x 转换为一个矩阵，包含三列：一列全是 1（用于截距），x 和 x^2。

```
def to_matrix(x):
    return np.vstack([np.ones(x.shape[0]),x,x**2]).T
```

（4）函数 E(x, y, theta)：用于计算预测值与实际值 y 之间的误差（损失）。使用均方误差（MSE）作为误差指标。

```
def E(x,y,theta):
    return 0.5 * np.sum((f(x,theta)) - data_y)**2
```

（5）函数 regression(x, y, theta)：使用梯度下降法拟合多项式模型到数据上。迭代更新 theta 以最小化误差。

```
def regression(x,y,theta):
    count=0
    diff=1
    new_x= to_matrix(standardize(x))
    error = E(new_x,y,theta)
    ETA = 1e-3
```

```
    while diff > 1e-4:
        theta = theta - ETA*np.dot(f(new_x,theta)-y,new_x)
        count+=1
        new_error = E(new_x,y,theta)
        diff = abs(new_error-error)
        error = new_error
        print('第{}次,theat={},差值={:.4f}'.format(count,theta,diff))
    return theta
```

（6）函数 getdata(path)：从 train_dataset.csv 文件路径加载训练数据。返回对应于特征和标签的数组 data_x 和 data_y。（为了区分 train 和 test，我写了两个数据读取函数）

```
def getdata(path):
    data_frame = pd.read_csv(r'D:\dataenclorse\first\train_dataset.csv')  # skiprows=14
    data_x,data_y = np.array(data_frame['x']), np.array(data_frame['y'])
    return data_x,data_y
```

（7）函数 getdata2(path)：从 test_dataset.csv 文件路径加载测试数据。返回测试特征数组 test_data_x 和 y。

```
def getdata2(path):
    data_frame = pd.read_csv(r'D:\dataenclorse\first\test_dataset.csv')  # skiprows=14
    test_data_x,y= np.array([data_frame['x'], np.array(data_frame['y'])])
    return test_data_x,y
```

3.具体实现

（1）初始化参数 theta，随机生成三个参数。

```
if __name__ == '__main__':
    theta = np.random.randn(3)
```

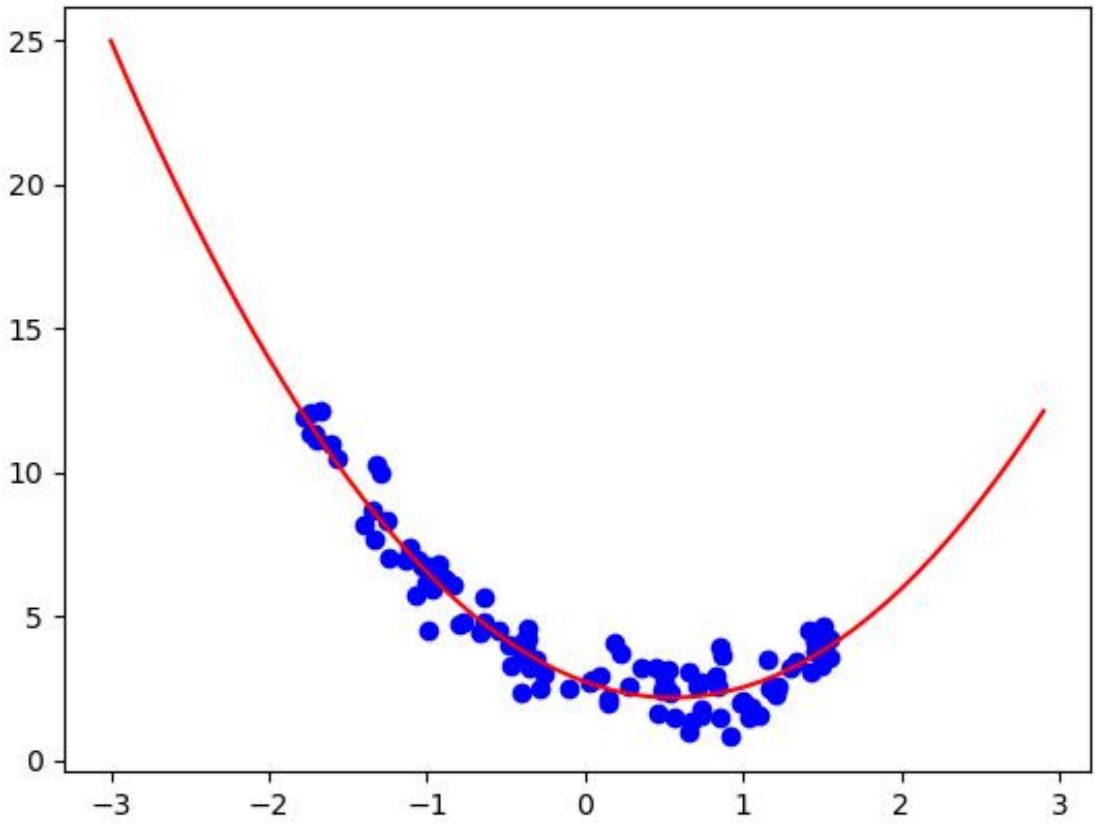
（2）加载训练数据，使用梯度下降法拟合模型。

```
data_x,data_y=getdata('train_dataset.csv')
    new_theta = regression(data_x,data_y,theta)
```

（3）标准化训练数据并绘制散点图，生成预测曲线并绘制。

```
 x = standardize(data_x)
    plt.scatter(x,data_y,c='blue')
    xx = np.arange(-3,3,0.1)
    plt.plot(xx,f(to_matrix(xx),new_theta),c='red')
    plt.show()
```

散点图如下：近似拟合为二次函数：



（4）加载测试数据，使用拟合的模型进行预测。

```
test_data_x,y=getdata2('test_dataset.csv')
test_data_y=f(to_matrix(standardize(test_data_x)),new_theta)
print(test_data_y)
```

展示结果：

```
[ 2.19372429  3.83959976  2.68234143  5.77086689  2.90147975  3.36182773
  4.1103422   2.42844503  3.27706043  6.5438024  13.71039261  2.17034139
  3.85644689  4.04752757  4.90115765  2.17728692 10.14278651  2.33715949
  3.19882011  2.17114805  9.18403809  2.29157175  8.98596315  3.70851268
  3.17825238]
```

4.将预测结果保存到指定路径的 CSV 文件中

```
file_path=r'D:\dataenclorse\first\t.csv'
with open(file_path,'w',newline='',encoding='utf-8') as f:
    fieldnames=['x','y']
    f_csv = csv.DictWriter(f, fieldnames=fieldnames)
    f_csv.writeheader()
    for i in range(0,len(test_data_y)):
        f_csv.writerow({'x':test_data_x[i],'y':test_data_y[i]})
pass
```

```
[ 2.19375029  3.83964114  2.68234234  5.77089327  2.9014727   3.36187131
  4.11029825  2.42848716  3.27710428  6.54382131 13.7103276   2.1703711
  3.85648818  4.04756773  4.90109233  2.17732097 10.14276558  2.33720023
  3.19880319  2.17117752  9.18402837  2.29159034  8.9859557   3.70855477
  3.17829649]
```

A	B	C
x	y	
0.4104237	2.1937243	
-0.121346	3.8395998	
0.6138093	2.6823414	
-0.342654	5.7708669	
0.6647234	2.9014797	
-0.048113	3.3618277	
0.8611648	4.1103422	
0.1647713	2.428445	
-0.033669	3.2770604	
-0.413492	6.5438024	
-0.890909	13.710393	
0.3701106	2.1703414	
-0.123722	3.8564469	
-0.149887	4.0475276	
0.9560719	4.9011576	
0.3157674	2.1772869	
-0.681213	10.142787	
0.2009698	2.3371595	
0.7228072	3.1988201	
0.3728106	2.171148	
-0.617151	9.1840381	
0.4801644	2.2915717	
-0.60338	8.9859632	
-0.102431	3.7085127	
-0.016116	3.1782524	